Empirical Convergence Analysis of Federated Averaging for Failure Prognosis ⋆

Maharshi Dhada * **Amit Kumar Jain** * **Ajith Kumar Parlikad** *

* *Department of Engineering, Institute for Manufacturing, University of Cambridge, Cambridge, CB3 0FS, UK*
correspondence e-mail (Maharshi Dhada) : mhd37@cam.ac.uk

Abstract: Data driven prognosis involves machine learning algorithms to learn from previous failures and generate its prediction model. However, often a single asset does not fail so frequently to have enough training data in the form of historical failures. This problem can be addressed by learning from failures across a cluster of similar other assets, but often working in different environments. The algorithm therefore must learn from a distributed dataset which might be heterogenous but with underlying similarities. Federated Learning is an emerging technique that has recently also been proposed as a fitting solution for prognosis of industrial assets. However, even the most commonly used Federated Learning algorithms lack theoretical convergence guarantees, and therefore their convergence must be analysed empirically. This paper empirically analyses the convergence of the Federated Averaging (FedAvg) algorithm for a fleet of simulated turbofan engines. Results demonstrate that while FedAvg is applicable for prognosis, it cannot acknowledge the differences in asset failure mechanisms. As a result, the prognosis framework needs to be modified such that similar failures are clustered together before FedAvg can be implemented.

Keywords: Prognosis, Machine Learning, Federated Learning, C-MAPSS

1. INTRODUCTION

Prognosis is critical for asset fleet upkeep. Traditionally, the underlying failure mechanisms had to be understood to formulate asset degradation using a mathematical function that mapped various asset characteristics to its approximate Remaining Useful Life (RUL) (Jardine and Tsang, 2005).

Advancements in the fields of sensors and computer science in recent decades have minimised the need for expert knowledge about failure mechanisms. Modern assets can instead be embedded with sensors measuring parameters such as temperature, vibrations, etc. at several internal and external locations. Their measurements over a period form time series of that asset's condition data. Segments of this time series corresponding to an asset's deteriorations from healthy states until failures are called failure trajectories (Kusiak, 2018).

Machine Learning (ML) algorithms can be trained using failure trajectories to generate a prediction model for that failure type. Such prediction models predict either an asset's RUL or its future health in real time based on the known failure modes (Kusiak, 2018). Using technologies such as Social Internet of Things (Li et al., 2018), ML algorithms can be deployed at individual asset level.

⋆ This research was funded by the EPSRC and BT Prosperity Partnership project: Next Generation Converged Digital Infrastructure, grant number EP/R004935/1

However, it is often found that a single asset fails so infrequently that enough data for the algorithm to generate a confident prediction model cannot be obtained (Voisin et al., 2013). In some cases however, training data comprising of failures distributed across several assets in a fleet, that may or may not be similar can be obtained. These assets might be operating in different conditions and environments which affect their rate of deterioration. The prognosis algorithms must therefore be capable of learning from a distributed non-IID (non-Independent and Identically Distributed) dataset with underlying similarities.

Collaborative prognosis (Palau et al., 2019a,b) offers a promising approach in this context, but requires assets to share data between each other to enable effective learning. However, the asset owners are likely to be competing organisations who would refrain from sharing their data with one another (Siemieniuch and Sinclair, 2002).

Similar to the above described problem of data driven prognosis is the problem faced by most mobile device applications. The plethora of ML algorithms working for applications like image classification or text predictions need to be trained using the data distributed across several devices. And since this data is personal to the users, it must not leave the individual devices (Konečnỳ et al., 2015).

Federated Learning (FL) was proposed as a solution to the problem of mobile device applications. FL technique forms a part of ML pipeline, where the prediction models learn from one another, rather than learning from the data

residing on the devices (Konečnỳ et al., 2015). Concretely, FL suits best for the applications with following characteristics:

- The training data is distributed across many devices, with unreliable or slow inter communication.
- The server does not have control over the working condition of the devices.
- The training data is non-IID, in the sense that the data lying on the individual devices are not samples from an overall common distribution. The data on each device has its own characteristic distribution which may or may not be similar to that of others.

Federated Averaging (FedAvg) is a primitive and the most widely used/ analysed FL algorithm yet. FedAvg has recently also been shown applicable for prognosis of asset fleets (Dhada et al., 2019), where it was implemented for training a neural network model using IID failure trajectories distributed across a fleet of turbofans.

However, FedAvg lacks theoretical guarantees for its convergence over a non-IID dataset (Li et al., 2019). The pioneering paper (McMahan et al., 2016) also provides empirical evidences based on certain image classification and text generation example applications for publicly available datasets only (Smith et al., 2017). The most recent example (Li et al., 2019) presents theoretical guarantees and conditions under which FedAvg converges for non-IID data, but the proofs are true only for convex optimisation problems. It is therefore useful to empirically analyse convergence of FedAvg for prognosis. The results presented in this paper shall highlight directions for future research, and enable the industries to know the expected accuracy of using FedAvg.

This paper discusses the results from an experiment where FedAvg was used to predict failures in a fleet of simulated turbofan engines. The experiment objective was to analyse and compare the predictions of neural networks trained using FedAvg for fleets with multiple and single failure modes, and with a neural network trained using the data directly. This was done for the case of IID (a single failure mode) and non-IID (multiple failure modes) datasets. Multiple hyper parameter settings of FedAvg were also tested. Existing literature advocates for reduced convergence rate for non-IID data compared to that for IID data. The results presented here show the extent of reduction in the convergence rate for prognosis of asset fleets. Moreover, difference in accuracy and certainty of predictions can also be inferred from the result plots.

A brief mathematical description of FedAvg is provided in Section 2. Detailed information about the same can be found in (McMahan et al., 2016; Konečnỳ et al., 2015). Section 3 explains the data used for the experiments, the framework for deploying FedAvg, and the various experiment cases conducted. Much of the content in Sections 2 and 3 is similar to (Dhada et al., 2019). In the Section 4 are presented the results from the corresponding experiments and lastly, interesting conclusions based on the results are discussed and extended to future research directions in Section 5.

2. FEDAVG MATHEMATICAL DESCRIPTION

FedAvg involves a server storing the global predictive model which is passed on to a randomly selected subset (or federation) of devices. Each device in the federation trains the model using its local data, and sends the parameter updates back to the server. The server aggregates these updates and generates a new global model (McMahan et al., 2016). The above steps constitute one round of communication, and it continues for several iterations until the global model converges. FedAvg steps while applying for a fleet of assets are schematically shown in Fig. 1, where steps 1 to 4 form one round of communication.

Consider a distributed system with K devices, total n data points across the whole system, and P_k being the set of indices of the data points on device k. The finite sum objective function $F(w)$ for the overall system can be written as:

$$F(w) = \sum_{k=1}^{K} (\frac{n_k}{n} * F_k(w)) \tag{1}$$

where $F_k(w)$ is the local objective function for the k^{th} device, and $n_k = |P_k|$ are the total number of data points on device k. The local objective functions may or may not be same as the global objective function.

Neural Networks commonly rely on Stochastic Gradient Descent to optimise the objective/ loss. FedAvg extends this for optimising objective (1).

Consider a federation of devices, of size C, selected in a given round of communication, and let s be the total number of data points contained in that federation. Each of the devices constituting the federation compute the average gradient $g_k = \nabla F_k(w_t)$ using their local data for the current global model parameters w_t. The server then aggregates the updates from the devices, and generates the updated global model for the next communication round as:

$$w_{t+1} \leftarrow w_t - \sum_{k=1}^{K} (\frac{n_k}{s} * g_k) \tag{2}$$

The averaging of model parameters of the commonly initialised neural network models described in (2) is empirically shown to converge, and that $\sum_{k=1}^{K} (\frac{n_k}{s} * g_k)$ is equivalent of $\nabla F(w_t)$. It enables the global model to learn from the data carried by each device, where the weight of the update corresponding to a given device is proportional to the amount of data carried by that device. This is the working principle of FedAvg.

Apart from the federation size C, the key governing parameters of FedAvg include the conventional neural network training parameters while training neural network at individual devices. For example: number of epochs per asset while evaluating g_k at the devices (EPA), local learning rate, and local batch size (McMahan et al., 2016)).

3. EXPERIMENTAL SETUP

Two fleets of 100 turbofan engines each were simulated using the "train_FD001" and "train_FD003" directories of the publicly available Commercial Modular Aero

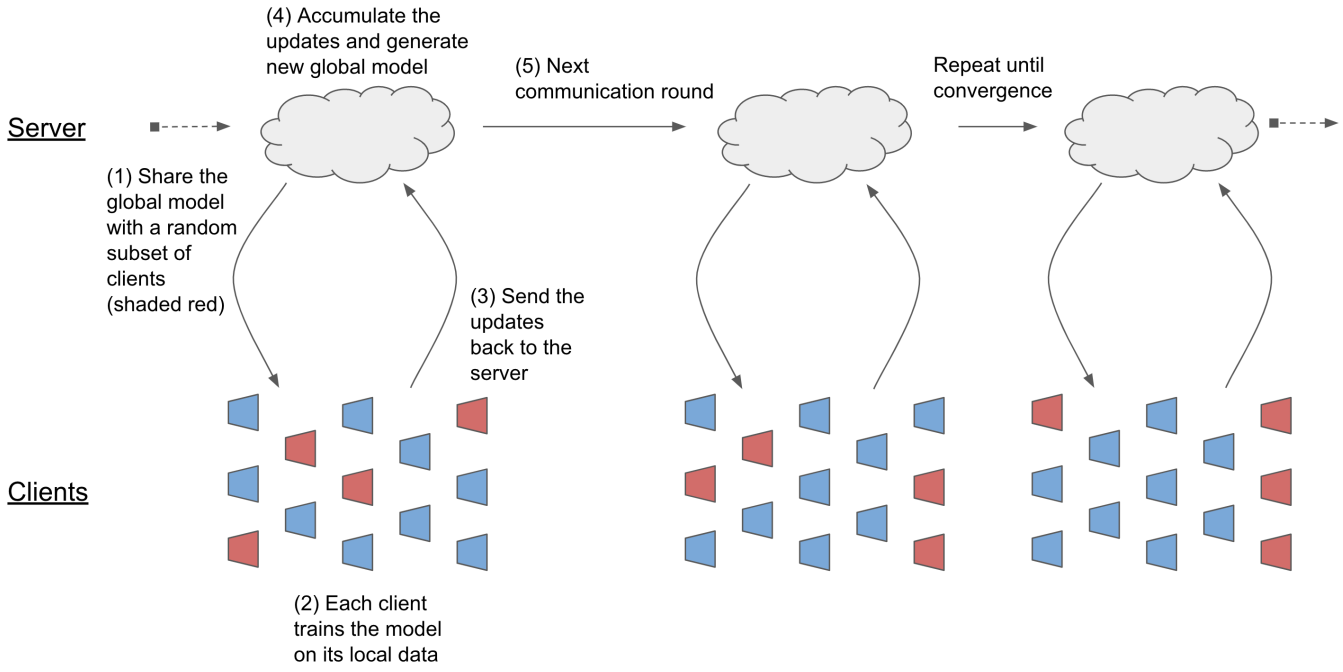

Fig. 1. Schematic representation of FedAvg for a client-server type networked system.

Propulsion System Simulation (C-MAPSS) dataset (Saxena et al., 2008)). Data for each asset was a single run-to-failure trajectory comprising of 24 features (3 representing the operating conditions, and the rest sensors). Therefore, the simulated fleet was equivalent to an industrial setting where the training data in the form of failure trajectories was uniformly distributed across 100 assets.

All 100 trajectories in train_FD001 belonged to the same failure type (high pressure compressor degradation) and operating conditions. However, train_FD003 fleet consisted of two failure modes (high pressure compressor degradation and turbine fan failure). Failure mode of any given asset in train_FD003 was not known a priori, therefore the failure trajectories constituting fleets train_FD001 and train_FD003 represented the IID and non-IID training datasets respectively.

The trajectories, corresponding to either failure modes, were multivariate time series. Some of the sensors however did not show any trends. It was believed that these sensors did not describe the asset's failure behaviour, and were therefore ignored to improve the training process. Concretely, the the sensors which showed standard deviations of less than 0.003 (after normalising the individual sensor values from 0 to 1) over the entire trajectory were ignored. Moreover, the noise associated with the sensor values was random. Such noise was filtered using rolling average (rolling average with window size 40). In summary, the trajectories were cleaned, the values normalised from 0 to 1, and the relevant sensors selected. After this preprocessing, train_FD001 transformed into a dataset consisting of 100 run-to-failures, with 17 features corresponding to each time step. Similarly, train_FD003 was transformed into a dataset consisting of 100 run-to-failures, and 18 features. The length of trajectories in either fleets varied across assets.

TensorFlow Federated (TFF) framework (Bonawitz et al., 2019)) was used to implement FedAvg for simulated fleets. This framework is an architecture similar to (Palau et al., 2019a)) used for collaborative prognosis. To draw an analogy, TFF represents "Digital Twins" as devices, and the "Social Platform" as server. Both the server and devices have three elements each: data repository, the analytics engine, and communications manager. These serve the purposes of storing the locally required data, analysing the local data, and sharing updates with the system respectively. Detailed information about the role played by these components can be found in (Bakliwal et al., 2018)). Each asset in the fleet was simulated and represented by a single instance of device. Keras library was used to train the neural networks at the asset level.

Drawing parallels with Section 2, the local updates in simulations were calculated by the devices. That is, the federation of assets selected at every communication round evaluated the updates for current neural network model using gradient descent. Following which, the server aggregated these updates according to (2).

A recurrent neural network (RNN, with one LSTM layer) was trained separately for each fleet using FedAvg, whose objective was to predict an asset's remaining useful life in real time, given the time series data until the current time step. Recurrent neural networks are a standard choice for predictive analytics of time series data. This is because of their flexibility in estimating the temporal relation between the features, and corresponding RUL of the asset (Palau et al., 2018)). The RNN used in the experiments

Table 1. Experiment cases

Experiment Case	C	EPA	Dataset
1.1	(5,10,20)	(1,1,1)	FD_001
1.2	(5,10,20)	(1,1,1)	FD_003
2.1	(5,5,5)	(1,5,10)	FD_001
2.2	(5,5,5)	(1,5,10)	FD_003
3 (same assets)	(5,5,10,5)	(1,5,5,10)	FD_001

described here comprised of four layers (following the input layer), where *tanh* activation functions were used across all neurons. The layers comprised of 10, 20, 5, and 1 neurons respectively, where the 10-neurons layer was the LSTM layer.

The experiments were aimed at analysing the effect of heterogeneous training data for different FedAvg parameter settings. Separate RNNs were trained for both fleets, and for the various values of parameters governing the learning process including (1) federation size (C), (2) Epochs per asset per communication round (EPA), and (3) using the same federation of assets for the communication rounds. Each of these parameters were varied during different experiment cases, while keeping the rest constant. The experiment case (3) corresponded to a situation where there was minimal variation in the failure trajectories. It resembled the expected performance of FedAvg for a fleet where the failure trajectories are all exactly similar.

Since the amount of data present at individual asset was very low, the effect of batch size was ignored. Moreover, the effect of learning rate while evaluating local updates can be found in (Dhada et al., 2019). Adam optimiser with a learning rate of 0.001 and Mean Absolute Error as the loss function were used for all experiment cases. The experiment cases and corresponding parameter values are indicated in the parentheses shown in table 1.

3.1 Performance Evaluation

In the experiments described here, the certainty of predictions was defined using a simple technique of neural networks ensembling. It involves deploying multiple neural networks with randomly initialised parameters for the same experiment case. This is similar to simulating the same experiment repeatedly, and thus enables us obtain a distribution of loss function values at each round of training/ communication. The mean of the distribution for a given communication round corresponds to the loss during that round, and the standard deviation to the certainty of prediction. Higher variance means lower certainty (Lakshminarayanan et al., 2017)).

In the experiments described here, 10 systems were simulated parallely for each experiment case. The neural network in each system was trained for 100 communication rounds (Except the case C=20 and EPA = 1 in Experiment 2, which was simulated for 75 rounds only). In the loss values vs. communication round plots presented in Section 4, the mean value of loss function at each communication round is shown using a bold line. The shaded region around this line is that of the first standard deviation. Lower values of the bold line mean that the neural network is more accurate and vice versa. Similarly, the tighter spread of the shaded region mean that the neural network is more certain about the predictions (Lakshminarayanan et al., 2017)).

To compare the performance of FedAvg with centralised training, the same neural networks were trained on the fleet data directly. That is: all 100 trajectories from each fleet were stored at a single compute node, and the neural networks trained on them. These networks were trained until they converged completely (when the loss function was no longer seemed to decrease with increasing epochs). Since the training was done on data directly, these results were assumed to be the best attainable performances with given neural network models.

4. RESULTS

Fig. 2 to 7 show plots of the results (average loss function values for the asset federation vs. communication rounds) obtained for experiment cases presented in Section 3.

Fig. 2 to 6 present the plots of loss function values for experiment cases (1) and (2) when RNN was trained using FedAvg. The plots in Fig. 7 correspond to case (3), where the same subset of assets failing in a single mode was chosen at every communication round. That is, a model is trained at the assets, the parameters averaged, and the new model is sent back to the same subset of fleet. The hyper parameter values for the corresponding case are mentioned in the plot titles.

Finally, Fig. 8 is the plot of loss function vs. number of epochs when the data from all the assets was stored in a common location and the neural network trained on the dataset directly. This corresponds to the traditional centralised prognosis approach used by the industries. Neural networks trained directly over the dataset are expected to be more accurate. This plot helps us understand the difference in accuracy encountered while moving to FedAvg for prognosis.

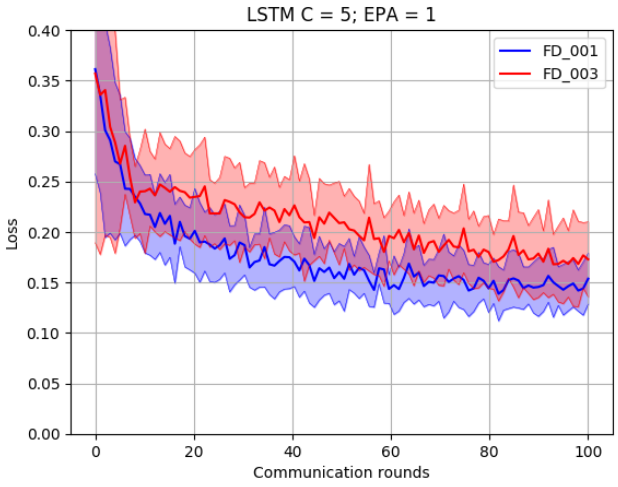

Fig. 2. Loss function values for experiment 1 and 2

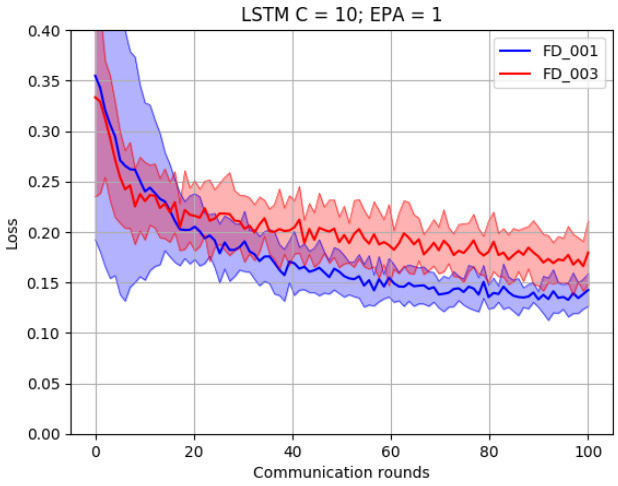

Fig. 3. Loss function values for experiment 1

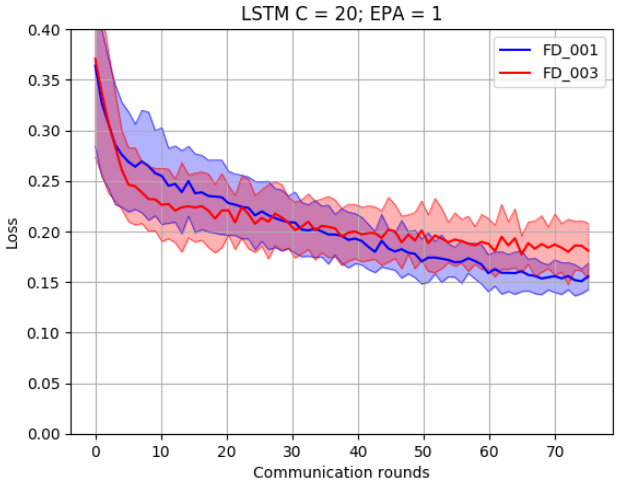

Fig. 4. Loss function values for experiment 1

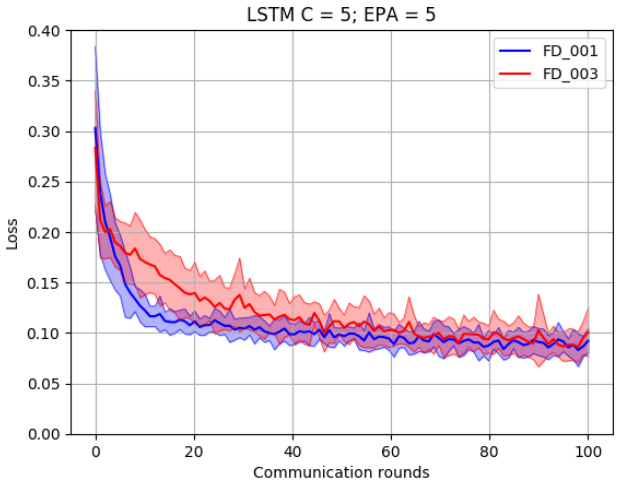

Fig. 5. Loss function values for experiment 2

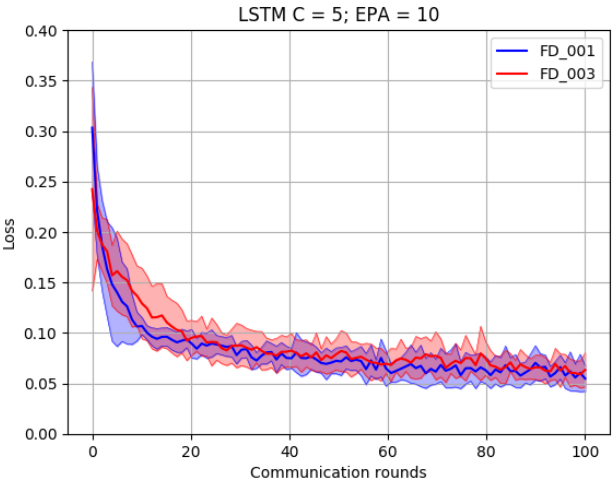

Fig. 6. Loss function values for experiment 2

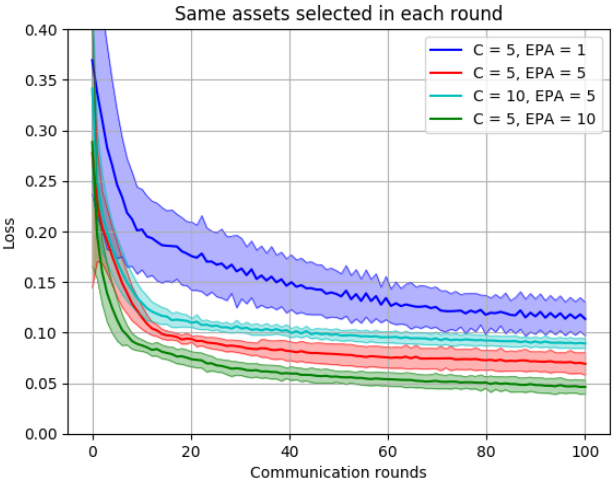

Fig. 7. Loss function values for experiment 3

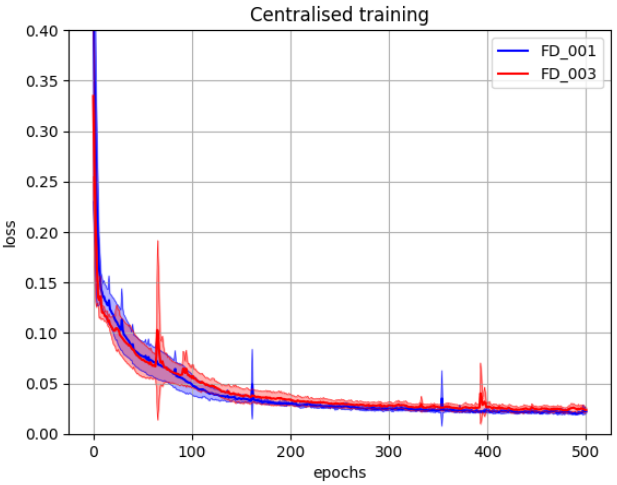

Fig. 8. Models trained using the fleet data directly

5. CONCLUSIONS AND FOLLOWING RESEARCH

This paper analyses the convergence of FedAvg for prognosis of non-IID failures in an asset fleet. In experiment 3, same federation of assets failing in a common failure mode was chosen at every communication round. Referring to Fig. 7, near optimal convergence in terms of accuracy and certainty could be obtained for a particular combination of C and EPA. Green plot in Fig. 7 corresponds to that combination. This observation substantiates the applicability of FedAvg for failure prognosis. However, the convergence tends to worsen when this subset of assets is replaced with a fleet of 100 assets with random subsets of assets selected at every round, as seen in Fig. 2 to 6. The authors believe this worsening is caused due to the heterogenous nature of asset fleets that the neural network is exposed to. Concretely, effect of two types of heterogeneities are identified from the plots:

(1) *Due to the presence of different failure modes:* The presence of multiple failure modes causes accuracy and certainty of the predictions to decline. Each of the figures 2 to 6 show that the loss encountered while training the network for FD_001 is lesser compared to FD_003. While in figure 6 the difference in loss values for either fleet is similar, FD_003 requires more communication rounds to the same value.

(2) *Due to the inter asset differences within those failing in the same mode:* This effect can be identified by comparing the convergence behaviour of FD_001 fleet for the cases where random subsets of assets are selected at each round (blue region in Fig. 2 to 6) with that when the same assets are selected (Fig. 7) for same hyper parameter settings. Confidence intervals are tighter, and the magnitude of loss lower for the case where the same subset of assets is selected. The authors believe this is because neural network depicted in Fig. 7 has to learn from a very limited range of failure behaviours, as opposed to several machines in a larger fleet.

It is therefore concluded that FedAvg, although being suitable for failure prognosis, is not capable of acknowledging the inter-asset differences within a fleet of assets. As a consequence, the performance of the prediction models worsen when the assets encounter different failure modes or when the fleet size increases. This is often the case in real world industries, and therefore a major limitation for implementing FedAvg in its current framework. Moreover, comparing the plots in Fig. 2 to 6 with one where the neural network model is trained directly over the data (Fig. 8) reveals the scope of improvement in Federated Learning. Fig. 8 shows the best performance that could be attained by the same neural networks for either fleet.

The authors understand FedAvg's underlying principle of averaging the parameters at each communication round to be the main bottleneck for this limitation. FedAvg algorithm does not facilitate any understanding of the underlying nature of the data, instead weighs the emanating trained models from the assets based on the quantity of data carried by the corresponding asset. Researchers in other fields have tried to address the problem of heterogeneity for Federated Learning by framing it as a multi-task learning problem (Smith et al., 2017).

REFERENCES

Bakliwal, K., Dhada, M.H., Palau, A.S., Parlikad, A.K., and Lad, B.K. (2018). A multi agent system architecture to implement collaborative learning for social industrial assets. *IFAC-PapersOnLine*, 51(11), 1237–1242.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H.B., et al. (2019). Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

Dhada, M., Salvador Palau, A., and Parlikad, A.K. (2019). Federated learning for collaborative prognosis. *COPEN'11, IIT Indore*.

Jardine, A.K. and Tsang, A.H. (2005). *Maintenance, replacement, and reliability: theory and applications*. CRC press.

Konečnỳ, J., McMahan, B., and Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*.

Kusiak, A. (2018). Smart manufacturing. *International Journal of Production Research*, 56(1-2), 508–517.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 6402–6413.

Li, H., Palau, A.S., and Parlikad, A.K. (2018). A social network of collaborating industrial assets. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 232(4), 389–400.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2019). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*.

Palau, A.S., Bakliwal, K., Dhada, M.H., Pearce, T., and Parlikad, A.K. (2018). Recurrent neural networks for real-time distributed collaborative prognostics. In *2018 IEEE international conference on prognostics and health management (ICPHM)*, 1–8. IEEE.

Palau, A.S., Dhada, M.H., Bakliwal, K., and Parlikad, A.K. (2019a). An industrial multi agent system for real-time distributed collaborative prognostics. *Engineering Applications of Artificial Intelligence*, 85, 590–606.

Palau, A.S., Liang, Z., Lütgehetmann, D., and Parlikad, A.K. (2019b). Collaborative prognostics in social asset networks. *Future Generation Computer Systems*.

Saxena, A., Goebel, K., Simon, D., and Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, 1–9. IEEE.

Siemieniuch, C. and Sinclair, M. (2002). On complexity, process ownership and organisational learning in manufacturing organisations, from an ergonomics perspective. *Applied Ergonomics*, 33(5), 449–462.

Smith, V., Chiang, C.K., Sanjabi, M., and Talwalkar, A.S. (2017). Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 4424–4434.

Voisin, A., Medina-Oliva, G., Monnin, M., Leger, J.B., and Iung, B. (2013). Fleet-wide diagnostic and prognostic assessment. In *Annual Conference of Prognostics and Health Management Society 2013*.